Informe de Diseño

Plataforma Unificada de Servicios Académicos (SOA)

Universidad Autónoma Veracruzana

Actividad: Arquitectura Orientada a Servicios

Fecha: Noviembre 22, 2025

Versión: 1.0

1. Introducción

La Universidad Autónoma Veracruzana enfrenta una crisis de integración tecnológica entre sus sistemas desacoplados. Este informe presenta el diseño e implementación de la Fase 1 de una plataforma web unificada basada en arquitectura orientada a servicios (SOA).

2. Análisis del Problema

2.1 Sistemas Existentes

• Sistema de Matrículas: SOAP parcialmente, base de datos MySQL local.

• Plataforma de Cursos Online: API REST aislada, sin integración.

• Sistema de Calificaciones: Base de datos independiente, exports manuales en múltiples formatos.

• Aplicación Móvil: Sin integración completa.

2.2 Desafíos Identificados

• Duplicación de datos entre sistemas

• Procesos manuales para sincronización

• Falta de interoperabilidad XML/JSON

• Escalabilidad limitada

3. Solución Propuesta: SOA

3.1 Principios de Diseño

1. **Independencia de Servicios**: Cada módulo es un servicio autónomo.
2. **Interoperabilidad**: Soporte de SOAP (XML) y REST (JSON).
3. **Escalabilidad**: Basado en microservicios, permite crecimiento horizontal.
4. **Reutilización**: APIs expuestas para múltiples consumidores.

3.2 Componentes de la Arquitectura

Servicio SOAP - Enrollments (Matrículas)

• Puerto: 5000

• Protocolo: SOAP 1.1 / XML

• Operaciones: GetEnrollments, CreateEnrollment

• Implementación: Python (Flask + lxml)

Servicio REST - Grades, Students, Courses

• Puerto: 5001

• Protocolo: HTTP REST / JSON

• Endpoints: /api/grades, /api/students, /api/courses

• Implementación: Python (Flask)

4. Modelo de Datos

Base de Datos MySQL (Railway)

• Host: shuttle.proxy.rlwy.net:22345

• Nombre: railway

• Tablas: students, courses, enrollments, grades

Tabla	Columnas	Relación
students	id, student_number, first_name, last_name, email	PK
courses	id, code, name, credits	PK
enrollments	id, student_id (FK), course_id (FK), enrolled_at, status	PK, FK
grades	id, enrollment_id (FK), grade, graded_at	PK, FK

5. Decisiones Técnicas

SOAP: Python + Flask + lxml (ligero, fácil de mantener, compatible con XML/SOAP)
REST: Python + Flask (consistencia con SOAP, simplicidad operacional)
Base de Datos: MySQL en Railway (acceso remoto, escalabilidad, compatibilidad)
Formatos: SOAP (XML) + REST (JSON) para interoperabilidad máxima
Autenticación: Ninguna en Fase 1 (implementar en Fase 2)

6. Casos de Uso

UC1: Consultar Matrículas (SOAP)

1. Sistema envía petición SOAP con student_id
2. Servicio consulta tabla enrollments
3. Retorna XML con enrollments del estudiante

UC2: Registrar Calificación (REST)

1. Portal envía POST a /api/grades
2. Servicio inserta en tabla grades
3. Retorna JSON con ID de nueva calificación

7. Plan de Implementación Posterior

Fase 2: Mejoras

• Autenticación y autorización (OAuth2, JWT)
• Validación de datos avanzada
• Logging y monitoreo
• Rate limiting
• Documentación de API (Swagger/OpenAPI)

Fase 3: Escalabilidad

• Caché (Redis)
• Balanceador de carga
• Contenedores (Docker)
• Orquestación (Kubernetes)
• API Gateway

8. Conclusiones

La arquitectura propuesta cumple con los requisitos de interoperabilidad, escalabilidad e integración unificada. La Fase 1 proporciona una base sólida para futuras expansiones sin necesidad de refactorización mayor.